

**Instituto Tecnológico de Costa Rica**

**Ingeniería en Computadores**

Profesor: Isaac Ramírez Herrera

Algoritmos y Estructuras de Datos 1

**Proyecto 1: “Space Invaders”**

Andrey Sibaja Garro

2017101898

2018

**Índice**

[Breve Descripción del Problema. 3](#_Toc511775751)

[Historias de Usuario: 3](#_Toc511775752)

[Historias de Usuario según Criticalidad y Secuencia de Uso 4](#_Toc511775753)

[Plan de Iteraciones 5](#_Toc511775754)

[Cronograma y Descomposición de Historias de Usuario en Tareas. 5](#_Toc511775755)

[Diagrama de Clases Inicial 7](#_Toc511775756)

[Diagramas de Secuencia 8](#_Toc511775757)

[Diagrama de Componentes 12](#_Toc511775758)

[Diagrama de Arquitectura 12](#_Toc511775759)

[Descripción de las estructuras de datos desarrolladas. 13](#_Toc511775760)

[Descripción Detallada de los Algoritmos Desarrollados 14](#_Toc511775761)

# Breve Descripción del Problema.

Se va a realizar un video juego llamado “Space Invaders” donde el jugador moviendo su nave principal horizontalmente en la parte inferior de la pantalla deba disparar a los enemigos que se estarán acercando poco a poco, esto con el fin de que los enemigos no lleguen donde se encuentra la nave. Existirán distintos tipos de hileras de enemigos así como jefes, donde el jugador deberá ir sobrepasando cada uno haciendo que estos nunca lleguen a la parte inferior.

# Historias de Usuario:

Como jugador quiero:

1. Que la nave se mueva horizontalmente en la parte inferior de la pantalla.
2. Que hayan enemigos en la parte superior de la pantalla.
3. Que los enemigos estén acomodados por hileras los cuáles se muevan hacia abajo y hacia los lados.
4. Que la nave deba eliminar los enemigos mediante disparos
5. Que la nave deba evitar que los enemigos lleguen a la parte inferior de la pantalla.
6. Que existan varios tipos de hileras.
7. Que la velocidad de los enemigos aumente conforme se avanza de nivel.
8. Que cada vez que se destruya un enemigo, estos se desplacen hacia el centro.
9. Que todos los enemigos sean iguales.
10. Que haya un jefe dependiendo el tipo de hilera, para destruir a este se requieren entre 2 y 5 disparos.
11. Que cuando se destruya el jefe se pase al siguiente nivel u otro tome su lugar (dependiendo de la hilera)
12. En la pantalla se deberá indicar el tipo de hilera actual y el próximo por salir, el nivel actual y el puntaje actual.
13. Que en un tipo de hilera los enemigos tengan distintas resistencias.
14. Poder controlar la nave principal con el teclado o mediante el acelerómetro del celular.
15. Que se muestre información del puntaje, tipo de hilera actual y la siguiente por venir en la pantalla del celular.

# Historias de Usuario según Criticalidad y Secuencia de Uso

Orden por secuencia de uso:

1, 2, 4, 3, 5, 8, 9, 6, 10, 11, 7, 13, 14, 15

A continuación se mostrará una matriz para asignar la criticalidad y secuencia de uso que van a tener las tareas establecidas en cada bloque de historias de uso. La clasificación de los colores para la matriz se explicará a continuación:

1. Color Rojo: representa el nivel de prioridad “Alto”, por lo que debe realizarse con mayor urgencia, y se debe considerar muy importante para el desarrollo del proyecto.
2. Color Amarillo: representa el nivel de prioridad “Medio”, por lo que es poco urgente pero importante.
3. Color Verde: representa el nivel de prioridad “Bajo”, por lo que es poco urgente y no tan importante.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Secuencia de Uso | | | |
| Criticalidad |  |  |  |  |
| 1, 2, 4 | 3, 5 |  |
|
| ↑ Minimal System Span ↑ | | | |
|  |  | 6, 8, 10, 7 | 11 |
|
|  |  | 9 | 13, 14, 15 |
|

Tabla 1. Matriz de Criticalidad y Secuencia de Uso de las Historias de Usuario

# Plan de Iteraciones

# Cronograma y Descomposición de Historias de Usuario en Tareas.

Bloque 1.

- Investigación e implementación de la librería gráfica LibGDX.

Bloque 2.

- Búsqueda de los sprites y el escenario para todos los elementos gráficos del juego.

- Hacer que la nave principal aparezca sobre el escenario.

- Hacer que la nave se mueva horizontalmente.

Bloque 3.

- Hacer que los enemigos aparezcan en la parte superior.

- Hacer que los enemigos tenga sus sprites en movimiento.

- Hacer que los enemigos se muevan hacía los lados y hacía abajo.

Bloque 4.

- Hacer que la nave dispare.

- Crear la lista simple con sus respectivos métodos.

- Crear el tipo de hilera básico.

- Cuando los enemigos son golpeados con un disparo que estos desaparezcan.

- Los enemigos se desplazan hacia el centro cuando se elimina uno de ellos.

Bloque 5.

- Crear el tipo de hilera A.

- Crear el jefe (aparece siempre en una posición aleatoria) y que este al ser eliminado se gane la ronda.

Bloque 6.

- Crear la lista doblemente enlazada con sus respectivos métodos.

- Crear el tipo de hilera B (El jefe cambia de posición aleatoria con los demás enemigos).

Bloque 7.

- Crear la lista circular con sus respectivos métodos.

- Crear el tipo de hilera C (Un enemigo aleatorio obtiene la posición de jefe cada que este es eliminado).

Bloque 8.

- Investigar sobre la implementación de “bubble sort”.

- Crear el tipo de hilera D (todos los enemigos tienen distintas resistencias, ordenados de izquierda a derecha, de menor a mayor resistencia).

Bloque 9.

- Crear la lista circular doblemente enlazada con sus respectivos métodos.

- Crear el tipo de hilera E (el jefe siempre está en el centro y la hilera gira en sentido reloj).

Bloque 10.

- Cuando los enemigos lleguen a la nave del jugador sea “game over”.

- Crear los distintos niveles.

- Que las hileras vengan al azar en cada nivel.

- Mostrar en pantalla el puntaje obtenido, el tipo de hilera actual y el tipo de hilera siguiente.

Bloque 11.

**-** Realizar la aplicación para el celular de manera que esta pueda controlar la nave mediante el acelerómetro.

# 

# Diagrama de Clases Inicial

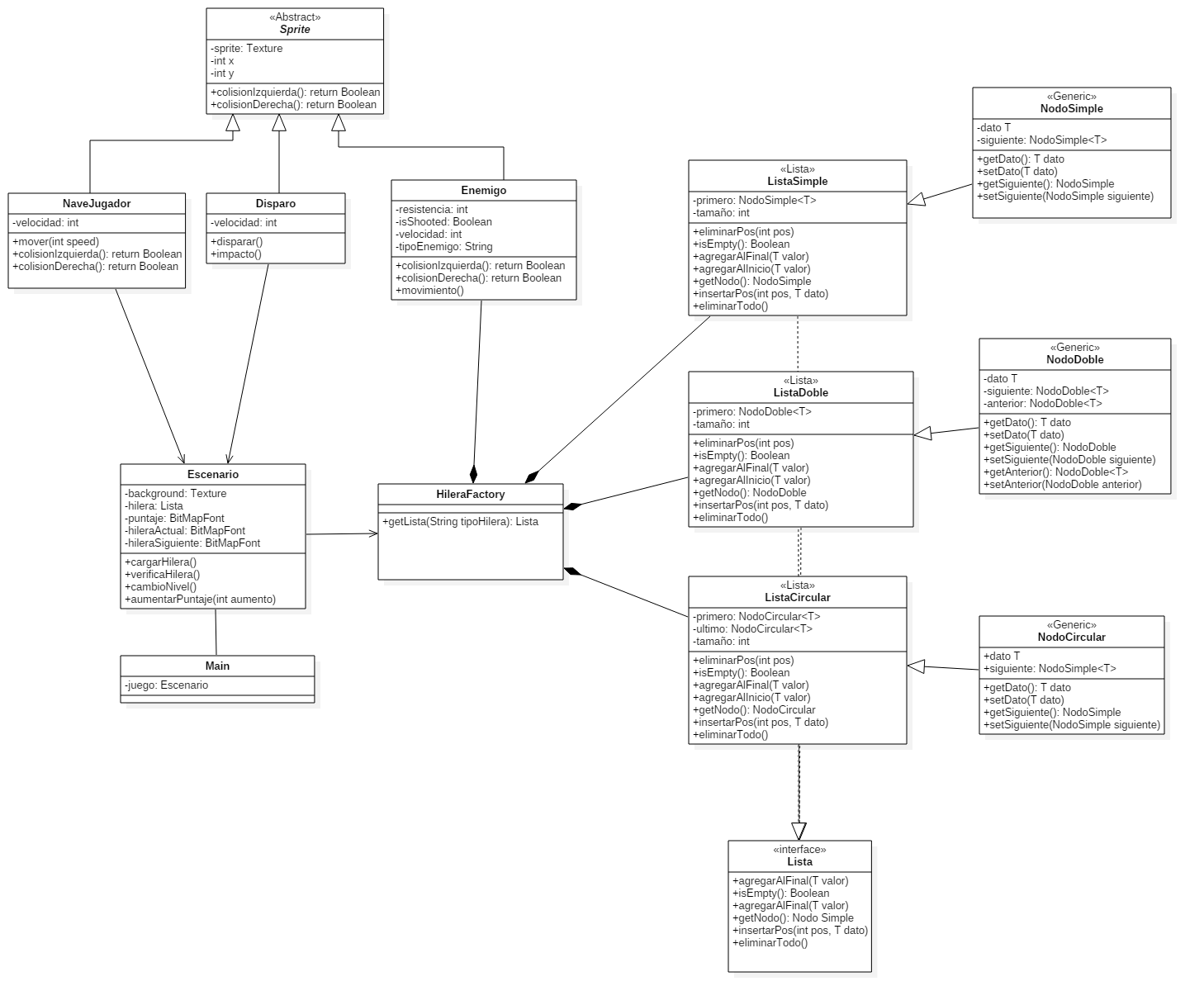


Figura 1. Diagrama de Clases Inicial

# 

# Diagramas de Secuencia

Historia de Usuario #1

**-** Que la nave se mueva horizontalmente en la parte inferior de la pantalla.

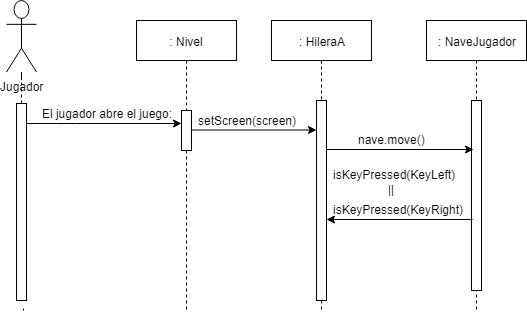
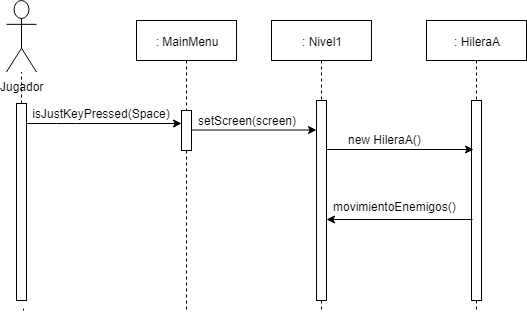


Figura 2. Diagrama de Secuencia de la primera historia de usuario.

El jugador al entrar en la pantalla del nivel, cada que presiona las teclas direccionales derecha o izquierda, se llamará el método en la clase “NaveJugador” donde hará que la nave se mueva hacia la izquierda o derecha según corresponda.

Historia de Usuario #3:

-Que los enemigos estén acomodados por hileras los cuáles se muevan hacia abajo y hacia los lados.



Nivel

Figura 3. Diagrama de Secuencia de la tercera historia de usuario.

El jugador entra y se topa con el menú donde si presiona la tecla “Space” entrará al nivel 1, este crea una nueva hilera y al ser creada se ejecuta el método que se encarga del movimiento hacia abajo y hacia los lados del enemigo.

Historia de Usuario #4:

- Que la nave deba eliminar los enemigos mediante disparos.

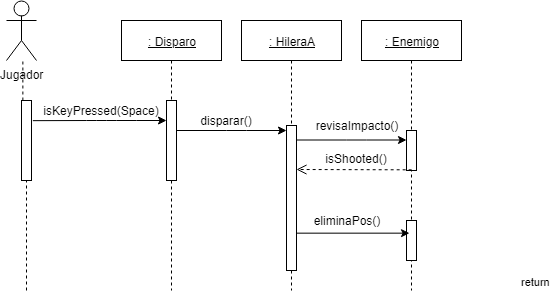


Figura 4. Diagrama de Secuencia de la cuarta historia de usuario.

El jugador presiona la tecla “Espacio” y este hace que el disparo se acomode justo frente de la nave y se mueva en dirección vertical hacia arriba, se realiza la validación si alguna nave fue impactada, si lo fue se cambia su atributo “shooted” a true y se procederá a eliminarse de la lista.

Historia de Usuario #5

- Que la nave deba evitar que los enemigos lleguen a la parte inferior de la pantalla.

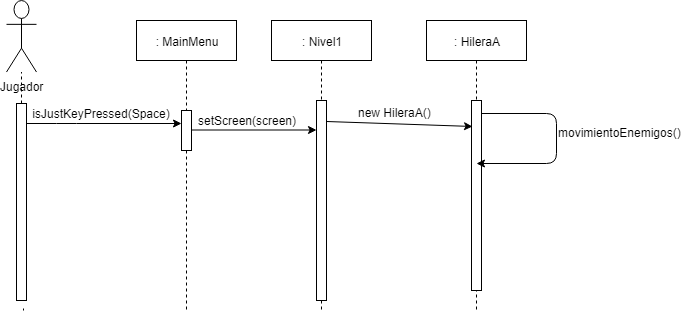


Figura 4. Diagrama de Secuencia de la quinta historia de usuario.

Cuando el jugador está en la pantalla de juego, cada vez que los enemigos se mueven se hace la validación si estos están a menos de una posición en el eje “y” y cuando se cumpla se dará por perdida la partida.

# Diagrama de Componentes

El siguiente diagrama presenta los paquetes donde se encuentra la lógica que el juego necesita para funcionar.

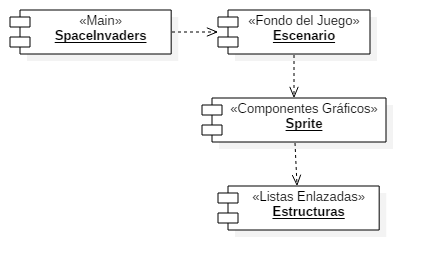


Figura 4. Diagrama de Componente del juego

# Diagrama de Arquitectura

El usuario mediante la interfaz gráfica LibGDX podrá tener acceso a todos los elementos gráficos que el juego presenta, esta es la encargada de la mayor parte del juego ya que es la que se encarga de dibujar todo aquello que el usuario espera. Detrás de todo esto está la lógica del juego y las estructuras utilizadas.

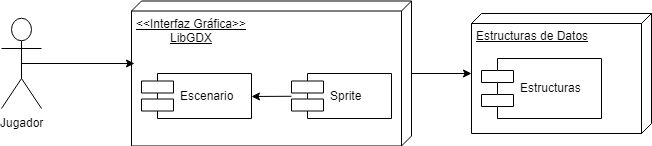
****

Figura 5. Diagrama de Arquitectura del juego

# Descripción de las estructuras de datos desarrolladas.

* **Listas Genéricas:** todas las listas implementas son genéricas, es decir, pueden guardar cualquier tipo de dato, en el caso de este proyecto se guardaron datos de tipo “Enemigo”.
* **Listas Enlazadas:** son estructuras de datos con operaciones definidas por el usuario, tiene sus ventajas sobre los ArrayList, como por ejemplo la capacidad de ir utilizando la memoria a medida que se va necesitando, los tipos de listas enlazadas utilizadas en este proyecto son:
* **ListaSimple:** es una estructura lineal en la que hay nodos entrelazados entre sí. Cada nodo tiene dos datos, uno de ellos es el valor que queremos almacenar y el otro es una referencia al siguiente nodo, esto hace que se forme la estructura de lista. Cuando uno de ellos apunta a “null” quiere decir que es el último nodo de la lista.
* **ListaDoble:** es una variación de la lista mencionada anteriormente, solo que esta vez los nodos tienen una nueva referencia que es al nodo anterior, es decir, podemos recorrer la lista hacia delante o hacia atrás.
* **ListaCircular:** en este tipo de lista todos los nodos tienen un “siguiente”. Al ser circular el primer elemento también es el último. Sus métodos son muy parecidos a los de las listas simples.
* **ListaCircularDoble:** es una mezcla entre la lista circular y la lista doble, podemos recorrer la lista hacia delante y hacia atrás sin llegar a un fin, ya que el primero es el último también.
* Las listas mencionadas anteriormente son utilizadas en los distintos tipos de hileras, mencionadas a continuación:

|  |  |
| --- | --- |
| **Hilera Básica** | ListaSimple |
| **Hilera A** | ListaSimple |
| **Hilera B** | ListaDoble |
| **Hilera C** | ListaCircular |
| **Hilera D** | ListaCircular |
| **Hilera E** | ListaCircularDoble |

Tabla 2. Listas implementadas en cada tipo de hilera.

# Descripción Detallada de los Algoritmos Desarrollados

Patrón de Diseño: Factory

Este patrón de diseño es utilizado para crear las listas enlazadas de los enemigos, o mejor dicho, las hileras que se van a ir mostrando en pantalla y que el jugador tiene que ir destruyendo.

La clase “ListaEnemigoFactory” es la que, mediante un String que recibe como parámetro, devuelve el tipo de hilera solicitado, a continuación se muestra el String que se requiere para cada hilera:

|  |  |
| --- | --- |
| **Hilera Básica** | “basica” |
| **Hilera A** | “claseA” |
| **Hilera B** | “claseB” |
| **Hilera C** | “claseC” |
| **Hilera D** | “claseD” |
| **Hilera E** | “claseE” |

Tabla 3. String solicitado por el Factory para la creación de las listas.

Bubble Sort

Es un algoritmo de ordenamiento, la manera en la que trabaja es revisando cada elemento de la lista e intercambiando las posiciones si estas son incorrectas, llevando siempre el mayor al final, lo hace varias veces hasta que la lista ya esté ordenada por completo.

En el proyecto este algoritmo es utilizado para la hilera D, donde los enemigos deben ir ordenados de mayor a menor resistencia en la pantalla de juego.